МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА



Отчет по лабораторной работе №2

по дисциплине

"Принципы и методы организации системных программных средств"

Тема работы: " Разработка программы ввода-вывода и обработки последовательности кодов на ассемблере "

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Викулова Е. Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Игнаков К. М.

19-В-2

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

**Цель работы:**

Приобретение навыков: разработки *одно*- и *много*сегментных программ на языке ассемблер, использования функций прерываний для организации ввода-вывода, управление трансляцией и компоновкой.

**Задание:**

**Часть 1.**

Написать программу на ассемблере, осуществляющую ввод последовательности символов с клавиатуры, обработку кодов символов в соответствии с заданием и вывод на экран результирующей последовательности.

8 *W*. Коды, не равные значениям *c*1, *c*2, *c*3.

**Часть 2.**   
Написать программу на ассемблере, осуществляющую вывод на экран даты создания BIOS (Aф=0FFFF5h) прямой записью в видеопамять (Аф=B8000h). Использовать точечные директивы (модель памяти, директивы сегментации).

**Теоретическая часть:**

Программа состоит из сегментов. Для описания сегментов можно использовать директивы SEGMENT / ENDS. Имена (всех) директив являются ключевыми словами языка, обрабатываются транслятором.

В односегментной программе все размещается в одном сегменте, также следует отступать 100h байт.

При работе с многосегментной и односегментной программой:  
tasm имя файла[.asm]

Многосегментная:

Tlink /v имя файла [.obj]

Односегментная:

Tlink /t имя файла [.obj]

**Директивы** **ассемблера** — параметры (ключевые слова) в тексте программы на языке **ассемблера**, влияющие на процесс ассемблирования или свойства выходного файла.

**Структура программы.**

Программа состоит из 3х сегментов, ASSUME связывает их имена с именами сегментных регистров. В программе выполняется настройка DS на начало сегмента данных. Программист определяет точку входа в программу END START. Настройка CS, SS производится автоматически на основании информации в заголовке .exe файла. Вывод строки на экран осуществляется вызовом 9-ой функции прерывания 21h DOS. Предварительно в DS помещается смещение адреса начала строки в сегменте данных. Для нормального завершения программы и выхода в операционную систему использован один из возможных вариантов, а именно вызов функции 4Ch прерывания INT 21h.

**Особенности работы с видеопамятью.**

Представляет собой область ОП, в которой находятся данные, которые используются для формирования изображения на экране монитора. Атрибуты символов - 1 байт, старшие четыре бита – цвет фона, младшие четыре бита – цвет символа.  
Видеопамять, текстовый режим - область памяти размером 64К. Диапазон адресов В800:0h-В800:0FFFFh. Каждый символ на экране – 2 байта в видеопамяти: младший байт ASCII - код символа; старший байт - атрибут символа. Позиция символа на экране определяется смещением в видеопамяти.

Код программ:

**Часть 1.**

Многосегментная программа.

d1 segment

mess\_enter\_data db 10, 13, 'Enter array: ', 10, 13, '$'

arr\_data dw 100 dup(?)

arr\_res dw 100 dup('$')

mess\_out\_res db 10, 13, 'Result: ', 10, 13, '$'

d1 ends

c1 segment

assume cs:c1, ds:d1

start:

mov ax, d1

mov ds, ax

mov dx, offset mess\_enter\_data

mov ah, 9 ; output data

int 21h

mov dx, offset arr\_data

mov arr\_data, 198 ; 2 bytes for max bytes and how much enter

mov ah, 10 ; read data

int 21h

mov si, offset arr\_data + 2

mov di, offset arr\_res

mov cx, [arr\_data + 1]

xor ch, ch

shr cx, 1 ; >> 2

m1:

mov ax, [si]

cmp ax, 3231h ; 12 in word

je m2

cmp ax, 3332h ; 23 in word

je m2

cmp ax, 3433h ; 34 in word

je m2

mov [di], ax

add di, 2

m2:

add si, 2

loop m1

m3: ; output result

mov dx, offset mess\_out\_res

mov ah, 9

int 21h

mov dx, offset arr\_res

mov ah, 9

int 21h

mov ah, 7

int 21h

mov ax, 4c00h

int 21h

c1 ends

end start

Односегментная программа.

c1 segment

assume cs:c1, ds:c1, es:c1, ss:c1

org 100h ; forever in one segment programs

start:

mov ah, 9

mov dx, offset mess\_enter\_data

mov ah, 9 ; output data

int 21h

mov dx, offset arr\_data

mov arr\_data, 198 ; 2 bytes for max bytes and how much enter

mov ah, 10 ; read data

int 21h

mov si, offset arr\_data + 2

mov di, offset arr\_res

mov cx, [arr\_data + 1]

xor ch, ch

shr cx, 1 ; >> 2

m1:

mov ax, [si]

cmp ax, 3231h ; 12 in word

je m2

cmp ax, 3332h ; 23 in word

je m2

cmp ax, 3433h ; 34 in word

je m2

mov [di], ax

add di, 2

m2:

add si, 2

loop m1

m3: ; output result

mov dx, offset mess\_out\_res

mov ah, 9

int 21h

mov dx, offset arr\_res

mov ah, 9

int 21h

mov ah, 7

int 21h

mov ax, 4c00h

int 21h

mess\_enter\_data db 10, 13, 'Enter array: ', 10, 13, '$'

arr\_data dw 100 dup(?)

arr\_res dw 100 dup('$')

mess\_out\_res db 10, 13, 'Result: ', 10, 13, '$'

c1 ends

end start

**Часть 2.**

c1 segment

assume cs:c1, ds:с1, es:c1, ss:c1

m1:

mov ax,3

int 10h

mov ax,0FFFFh ; START DATA OF BIOS

mov ds,ax

mov ax,0B800h

mov es,ax

mov si,5

mov di,1690

mov cx,8

mov ah,0C2h

m2:

mov al,[si]

mov es:[di],ax

inc di

inc di

inc si

loop m2

mov ax,4C00h

int 21h

c1 ends

end m1

**Файл .lst**

Turbo Assembler Version 3.1 30/11/21 15:56:14 **Page 1**

2\_lab.asm

1 0000 d1 segment

2 0000 0A 0D 45 6E 74 65 72+ mess\_enter\_data db 10, 13, 'Enter array: ', 10, 13, '$'

3 20 61 72 72 61 79 3A+

4 20 0A 0D 24

5 0012 64\*(????) arr\_data dw 100 dup(?)

В первом столбце записано смещение относительно начала сегмента, во втором значения, которыми проинициализированы переменные. Для строки mess\_enter\_data указана кодировка каждого символа в ASCII таблице. Для переменной arr\_data указано каким символов она проинициализирована и количество таких символов.

6 00DA 64\*(0024) arr\_res dw 100 dup('$')

7 01A2 0A 0D 52 65 73 75 6C+ mess\_out\_res db 10, 13, 'Result: ', 10, 13, '$'

8 74 3A 20 0A 0D 24

9 01AF d1 ends

10

11 0000 c1 segment

12 assume cs:c1, ds:d1

13

14 0000 start:

15 0000 B8 0000s mov ax, d1

16 0003 8E D8 mov ds, ax

17 0005 BA 0000r mov dx, offset mess\_enter\_data

18 0008 B4 09 mov ah, 9 ; output data

19 000A CD 21 int 21h

20

21 000C BA 0012r mov dx, offset arr\_data

22 000F C7 06 0012r 00C6 mov arr\_data, 198 ; 2 bytes for max bytes and how much enter

Во втором столбце – КОП (код операции), в третьем – информация об операндах. Для команд, в которых используются переменные (d1 и e1) указано смещение переменных относительно начала их сегментов. В командах, использующих переменные, объявленные внутри сегмента данных в третьем столбце указано их смещение. Заносим данные в переменную, 1 байт показывает максимальную длину введенного массива, 2 байт реальную информацию, сколько ввел пользователь.

23 0015 B4 0A mov ah, 10 ; read data

24 0017 CD 21 int 21h

25

26 0019 BE 0014r mov si, offset arr\_data + 2

Со второго байта входного массива начинаются данные. Offset – нужно обрабатывать переменную по имени.

27 001C BF 00DAr mov di, offset arr\_res

28

29 ; mov ax, cs

30 ; add ax, 100

31 ; mov ds, ax

32 ; mov si, ax

33 ; add ax, 100

34 ; mov es, ax

35 ; mov di, ax

36

37 001F 8B 0E 0013r mov cx, [arr\_data + 1]

38 0023 32 ED xor ch, ch

39 0025 D1 E9 shr cx, 1 ; >> 2

40

41 0027 m1:

42 0027 8B 04 mov ax, [si] ; Change

43

44 0029 3D 3231 cmp ax, 3231h ; 12 in word

45 002C 74 0F je m2

46

47 002E 3D 3332 cmp ax, 3332h ; 23 in word

48 0031 74 0A je m2

49

50 0033 3D 3433 cmp ax, 3433h ; 34 in word

51 0036 74 05 je m2

52

53 0038 89 05 mov [di], ax

54

55 003A 83 C7 02 add di, 2

56

57 003D m2:

Turbo Assembler Version 3.1 30/11/21 15:56:14 **Page 2**

2\_lab.asm

58 003D 83 C6 02 add si, 2

59

60 0040 E2 E5 loop m1

61

62 0042 m3: ; output result

63 0042 BA 01A2r mov dx, offset mess\_out\_res

64 0045 B4 09 mov ah, 9

65 0047 CD 21 int 21h

66

67 0049 BA 00DAr mov dx, offset arr\_res

68 004C B4 09 mov ah, 9

69 004E CD 21 int 21h

70

71 0050 B4 07 mov ah, 7

72 0052 CD 21 int 21h

73

74 0054 B8 4C00 mov ax, 4c00h

75 0057 CD 21 int 21h

76

77 0059 c1 ends

78 end start

Turbo Assembler Version 3.1 30/11/21 15:56:14 **Page 3**

Symbol Table

Symbol Name Type Value

??DATE Text "30/11/21"

??FILENAME Text "2\_lab "

??TIME Text "15:56:14"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text C1

@FILENAME Text 2\_LAB

@WORDSIZE Text 2

ARR\_DATA Word D1:0012

ARR\_RES Word D1:00DA

M1 Near C1:0027

M2 Near C1:003D

M3 Near C1:0042

MESS\_ENTER\_DATA Byte D1:0000

MESS\_OUT\_RES Byte D1:01A2

START Near C1:0000

Groups & Segments Bit Size Align Combine Class

C1 16 0059 Para none

D1 16 01AF Para none

**Файлы \*.map**

Атрибут выравнивания сегмента сообщает компоновщику о том, что нужно обеспечить размещение начала сегмента на заданной границе.

## Выравнивание byte

Выравнивание не выполняется, то есть следующий сегмент расположен сразу после предыдущего.

Start Stop Length Name Class

00000H 0001FH 00020H D1 DATA

00020H 00040H 00021H E1 DATA

00041H 00108H 000C8H ST1 STACK

00109H 0016DH 00065H C1 CODE

Program entry point at 0010:0009

## Выравнивание word

Сегмент начинается по адресу, кратному двум, т. е. последний (младший) значащий бит физического адреса равен 0 (выравнивание на границу слова).

Start Stop Length Name Class00000H 000E2H 000E3H D1 000E4H 0014CH 00069H C1 Program entry point at 000E:0004Warning: No stack

## Выравнивание para

Сегмент начинается по адресу, кратному 16d(10h).

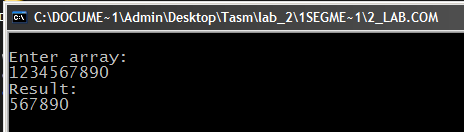
Start Stop Length Name Class  
00000H 000E2H 000E3H D1   
000F0H 00158H 00069H C1   
Program entry point at 000F:0000  
Warning: No stack

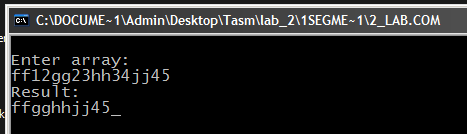
## Выравнивание page

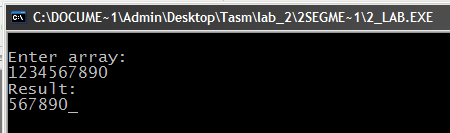
Сегмент начинается по адресу, кратному 256d (100h).

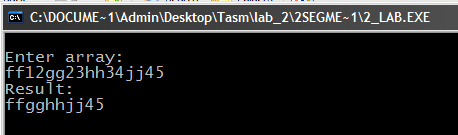
Start Stop Length Name Class  
00000H 000E2H 000E3H D1   
00100H 00168H 00069H C1   
Program entry point at 0010:0000  
Warning: No stack

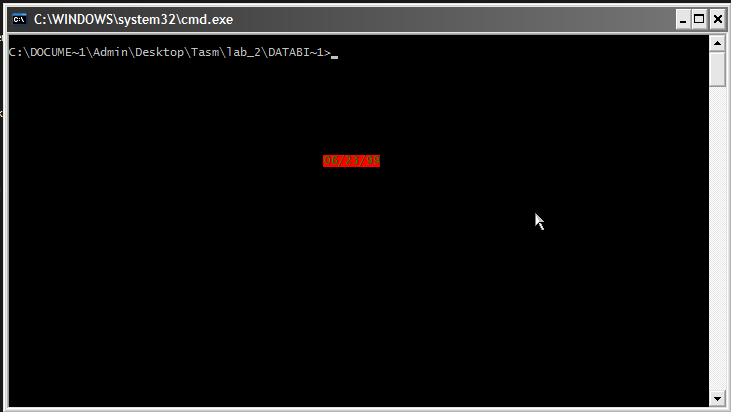
**Результат выполнения:**

****

****

****

****

****

**Структура файлов**

Файл EXE, создаваемый компоновщиком, состоит из двух частей:

* управляющая информация для загрузчика;
* загрузочный модуль.

Информация для загрузчика, описанная ниже, расположена в начале файла и образует так называемый заголовок. Сразу за ним следует тело загрузочного модуля, представляющее собой копию образа памяти задачи, построенной компоновщиком.

Стандартная часть заголовка имеет следующий формат:

* **00-01** 4D5A — сигнатура файла .EXE;
* **02-03** Длина образа задачи по модулю 512 (то есть число полезных байт в последнем блоке). Компоновщики версий до 1.10 помещали в это поле 04; если оно имеет такое значение, его рекомендуется игнорировать);
* **04-05** Длина файла в блоках;
* **06-07** Число элементов таблицы настройки адресов;
* **08-09** Длина заголовка в 16-байтных параграфах. Используется для выяснения начала тела загрузочного модуля;
* **0A-0B** Минимальный объём памяти, которую нужно выделить после конца образа задачи (в 16-байтных параграфах);
* **0C-0D** Максимальный объём памяти, которую нужно выделить после конца образа задачи (в 16-байтных параграфах);
* **0E-0F** Сегментный адрес начала стекового сегмента относительно начала образа задачи;
* **10-11** Значение SP при входе в задачу;
* **12-13** Контрольная сумма — ноль минус результат сложения без переноса всех слов файла;
* **14-15** Значение IP (счетчика команд) при входе в задачу;
* **16-17** Сегментный адрес начала кодового сегмента относительно начала образа задачи;
* **18-19** Адрес первого элемента таблицы настройки адресов относительно начала файла;
* **1A-1B** Номер сегмента перекрытий (0 для корневого сегмента программы).

Далее следует таблица настройки адресов. Таблица состоит из элементов, число которых записано в байтах 06-07. Элемент таблицы настройки состоит из двух полей: 2-байтного смещения и 2-байтного сегмента, и указывает слова в загрузочном модуле, содержащее адрес, который должен быть настроен на место памяти, в которое загружается задача. Настройка производится следующим образом:

1. В области памяти после резидентной части выполняющей загрузку программы строится префикс программного сегмента (PSP);
2. Стандартная часть заголовка считывается в память;
3. Определяется длина тела загрузочного модуля (разность длины файла 04-07 и длины заголовка 08-09 плюс число байт в последнем блоке 02-03). В зависимости от признака, указывающего загружать задачу в конец памяти или в начало, определяется сегментный адрес для загрузки. Этот сегмент называется начальным сегментом;
4. Загрузочный модуль считывается в начальный сегмент;
5. Таблица настройки порциями считывается в рабочую память;
6. Для каждого элемента таблицы настройки к полю сегмента прибавляется сегментный адрес начального сегмента. В результате элемент таблицы указывает на слово в памяти, к которому прибавляется сегментный адрес начального сегмента;
7. Когда таблица настройки адресов обработана, в регистры SS и SP записываются значения, указанные в заголовке, а к SS прибавляется сегментный адрес начального сегмента. В ES и DS записывается сегментный адрес начала PSP. Управление передается по адресу, указанному в заголовке (байты 14-17).

**Структура программного сегмента**

При обращении к нерезидентной команде или вызове программы операцией Exec, DOS определяет минимальный адрес, начиная с которого может быть загружена соответствующая программа. Эта область называется программным сегментом.

По смещению 0000 в программном сегменте DOS формирует префикс программного сегмента (PSP). Сама программа загружается по смещению 0100.

Программа завершается переходом по адресу 0000 в программном сегменте, выполнив INT 20, или выполнив INT 21 с AH=0 или AH=4C, или обратившись к подпрограмме по адресу 0050 в программном сегмент с AH=0 или AH=4C.

Примечание: при завершении иначе, чем операцией 4C, программа должна предварительно заслать в CS адрес начала своего программного сегмента.

Все четыре способа возвращают управление в резидентную часть COMMAND.COM (при этом операция 4C передает код завершения). Все четыре способа приводят к продолжению выполнения программы, обратившейся к операции Exec (4B). При этом вектора прерываний 22, 23 и 24 (завершение, Ctrl-Break, фатальная ошибка обмена) восстанавливаются из Префикса Программного сегмента возобновляемой задачи. Затем управление передается по адресу завершения. Если программа возвращается в COMMAND.COM, то управление передается в нерезидентную часть. Если это происходит во время выполнения командного файла, оно продолжается, иначе COMMAND выдает на терминал приглашение и ждет ввода следующей команды.

Когда загруженная программа получает управление, имеют место следующие условия:

Для всех программ:

* 1. В префиксе программного сегмента по смещению 2C передается адрес среды. Среда представляет собой последовательность строк ASCIIZ, вида параметр=значение. Общая длина строк среды не более 32 Кбайт; среда начинается с границы параграфа. После последней строки следует нулевой байт. Среда, передаваемая задаче от COMMAND, содержит, как минимум, параметр COMSPEC=(значение этого параметра — полное имя файла, содержащего используемый COMMAND.COM). Она также содержит значения, установленные командами PATH, PROMPT и SET. Передаваемая среда является копией среды родительского процесса. Если задача остается резидентом, то последующие команды PATH, PROMPT и SET не будут воздействовать на её среду.
  2. По смещению 0050 в префиксе программного сегмента содержится программа обращения к операциям DOS. Таким образом, занеся в AH номер операции, программа может вызвать процедуры (LCALL) по адресу PSP + 50, а не обращаться к прерыванию 21.
  3. Адрес буфера DTA установлен на PSP +80.
  4. Блоки управления файлами, расположенные по смещениям 5C и 6C в префиксе программного сегмента заполняются в соответствии с параметрами командной строки. При этом если соответствующий параметр включает имя каталога, в FCB заносится только код устройства, имя файла формируется неправильно.
  5. Неформатная часть, начинающаяся со смещения 81, содержит символы командной строки после имени команды, включая все пробелы и разделители. По смещению 80 помещена длина этой строки. Если командная строка включает параметры переназначения (на них указывают символы > и <) они не попадают сюда, так как переназначение прозрачно для программ.
  6. Слово по смещению 6 содержит число байт в данном сегменте.
  7. Регистр AX указывает, правильно ли заданы имена устройств в параметрах:
  8. AL = FF — имя устройства для первого параметра задано неверно, иначе AL = 00;
  9. AH = FF — имя устройства для первого параметра задано неверно, иначе AH = 00.

Для программ .EXE:

* 1. DS и ES указывают на начало префикса программного сегмента.
  2. Регистры CS, IP, SS и SP получают значения, указанные компоновщиком.

Для программ .COM:

* 1. Все четыре сегментных регистра указывают на префикс программного сегмента.
  2. Программе выделяется вся свободная память. Если программа запускает другие программы операцией Exec, то она должна освободить для неё часть памяти операцией Setblock (4A)
  3. Счетчик команд IP получает значение 0100H.
  4. Регистр SP указывает на конец программного сегмента. Длина сегмента в ячейке 6 префикса уменьшается на 0100H, чтобы освободить пространство для стека такого размера.
  5. На вершину стека помешается нулевое слово.

**Вывод:**

В ходе лабораторной работы я приобрел навыки разработки одно- и многосегментных программ на языке ассемблер, использования функций прерываний для организации ввода-вывода, управление трансляцией и компоновкой.